SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TADDM – SERVICENOW-INTEGRAATIO

Palvelinten välisten relaatiotietojen siirtäminen TADDM-ohjelmasta ServiceNow-alustaan

TEKIJÄ: Rene Silfsten

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Rene Silfsten	
Työn nimi TADDM – ServiceNow integraatio	
Päiväys 24.2.2015	Sivumäärä/Liitteet 38 / 2
Ohjaaja(t) lehtori Keijo Kuosmanen, lehtori Jussi Koistinen	
Toimeksiantaja/Yhteistyökumppani(t) Enfo Oyj, laatu- ja prosessijohtaja Marko Ikäheimo	

Tiivistelmä

Opinnäytetyössä tehtiin integraatio IBM:n Tivoli Application Dependency Discovery Manager -konfiguraationhallintaohjelman ja ServiceNow-palvelunhallinta-alustan välille. IBM:n TADDM-ohjelmasta siirrettiin ServiceNow-alustaan palvelinten väliset relaatiotiedot. Työ tehtiin Enfo Oyj:lle yrityksessä käynnissä olleen ”Konfiguraation- ja muutoksenhallinnan työkalu–POC (TADDM)” –nimisen projektin yhteydessä.

Työ aloitettiin perehtymällä sekä palvelunhallintaan että konfiguraationhallintaan ja TADDM-ohjelmaan. Enfo Oyj:n käyttämä ServiceNow-palvelunhallinta-alusta oli tullut työn puolesta tutuksi hiukan yli vuoden työssäolon aikana. Seuraavaksi selvitettiin, mitä tietoja TADDM-ohjelma sisälsi, miten tietoja saataisiin luettua ja kuinka tietojen siirtäminen ServiceNow-alustaan tulisi tehdä Enfo Oyj:n yleisesti käyttämiä integraatiokanavia hyödyntäen. Lopuksi toteutettiin suunniteltu integraatio.

Työn lopputuloksena oli toimiva integraatio TADDM–ohjelman ja ServiceNow-alustan välillä. Vaikka opinnäytetyössä siirrettiin TADDM-ohjelmasta ainoastaan palvelinten väliset relaatiotiedot, Enfo Oyj voi samaa periaatetta ja integraatiokanavaa käyttäen siirtää käytännössä mitä tahansa TADDM-ohjelman sisältämää tietoa ServiceNow-alustaan. Opinnäytetyö sisältää myös tiiviin katsauksen palvelunhallinnasta ja konfiguraationhallinnasta.

Avainsanat
palvelunhallinta, konfiguraationhallinta, TADDM, ServiceNow, ITIL

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Rene Silfsten			
Title of Thesis TADDM ServiceNow Integration			
Date	24 February 2015	Pages/Appendices	38 / 2
Supervisor(s) Mr. Keijo Kuosmanen, Lecturer and Mr. Jussi Koistinen, Lecturer			
Client Organisation /Partners Enfo Oyj, Mr. Marko Ikäheimo, Quality and Process Director			

Abstract

The purpose of this thesis was to make an integration between IBM's Tivoli Application Dependency Discovery Manager configuration management software and ServiceNow service management platform. Relationship information between servers was transferred from the IBM's TADDM software to the ServiceNow platform. The thesis was made for Enfo Oyj as a part of the "Konfiguraation- ja muutoksenhallinnan työkalu-POC (TADDM)" project that was in progress at Enfo Oyj at the time.

The thesis was started by familiarizing oneself with service management, configuration management and the TADDM software. The ServiceNow service management platform had already become familiar during a little over a year's employment relationship.

Next, the contents of the TADDM software were studied, a way to read the data from the TADDM software was figured out and an integration method using Enfo Oyj's standard integration channels was researched. Finally, the integration was executed.

The end result was a working integration between the TADDM software and the ServiceNow platform. Although only the relationship information between servers was transferred during the project, Enfo Oyj can utilize the same integration method to transfer virtually any information from the TADDM software to the ServiceNow platform. The thesis also contains a compact review of service management and configuration management.

Keywords
service management, configuration, TADDM, ServiceNow, ITIL

ESIPUHE

Kiitokset Enfo Oyj:lle ja laatu- ja prosessijohtaja Marko Ikäheimolle opinnäytetyön aiheesta. Kiitokset myös Tero Savolaiselle, Eero Kuutschinille ja Ismo Airaksiselle neuvoista, joita työn edetessä heiltä sain, sekä Tarja Hakkaraiselle ITIL-kirjaston ynnä muun materiaalin lainaamisesta.

Haluan myös kiittää erityisesti puolisoani ja perhettäni kannustuksesta ja tuesta.

Kuopiossa 24.2.2015
Rene Silfsten

SISÄLTÖ

LYHENTEET JA MÄÄRITELMÄT

API = Application Programming Interface. Rajapinta, minkä kautta pääsee käsiksi esimerkiksi TADDM:n sisältämiin tietoihin (IBM 2014b).

ASSET = Mikä tahansa palveluntarjoajan resurssi tai kyky, joka edistää palvelun tuottamista (Office of Government Commerce 2010e, 226).

CDM = Common Data Model. Relaatiotietokantaan pohjautuva, objekteista koostuva hierarkkinen tallennusmalli, jota TADDM käyttää tietojensa tallentamiseen (IBM 2014c).

CI = Configuration Item eli konfiguraatio-objekti. Mikä tahansa esine tai asia mitä täytyy hallita jotta voidaan toimittaa jokin IT-palvelu. Jokaisen konfiguraatio-objektin eli CI:n tiedot tallennetaan konfiguraatiotietueeseen, joka on konfiguraationhallintajärjestelmässä. Objekteihin kuuluvat tyypillisesti esimerkiksi laitteistot, ohjelmat, rakennukset, ihmiset ja dokumentaatiot. (Office of Government Commerce 2010e, 230.)

CMDB = Configuration Management Database. Tietokanta, jossa säilytetään konfiguraatio-objektien tiedot ja relaatiot muihin objekteihin. CMS (Configuration Management System) ylläpitää yhtä tai useampaa tällaista tietokantaa. (Office of Government Commerce 2010e, 230.)

CMS = Configuration Management System. Konfiguraationhallintajärjestelmä eli sarja työkaluja ja tietokantoja, joiden avulla hallitaan konfiguraatiotietoja. Järjestelmän työkalujen avulla voidaan kerätä, tallentaa, hallita, päivittää ja esittää CI tietoja ja niiden välisiä relaatioita. Sisältää yhden tai useamman CMDB:n. (Office of Government Commerce 2010e, 230.)

Integraatio = prosessi, jossa määritetään yhteys kahden sovelluksen ja/tai verkkopalvelun välille (SerivceNow 2014b).

IT = Information technology. Informaatioteknologia. (Gaughan, McWhirter 2012, 1.)

ITIL = IT Infrastructure Library. Palvelunhallinnan lähestymistapa. Käsite on luotu 1980 – luvun loppupuolella Iso-Britannian hallituksen aloitteesta, minkä tavoitteena oli luoda tehokas, menestyvä ja luotettava lähestymistapa palvelunhallintaan. (Office of Government Commerce 2007, 3.)

ITSM = IT Service Management. IT-palvelunhallinta. (Office of Government Commerce 2007, 3.)

JSON = tekstiformaatti, jota käytetään jäsennetyn datan vaihtamiseen kaikkien ohjelmointikielien välillä (Ecma international, 2013).

MQL = Model Query Language. IBM:n TADDM-ohjelman käyttämä kyselykieli (IBM 2014j).

OGC = Office of Government Commerce. Iso-Britannian hallituksen osasto, joka omistaa ITIL-tavaramerkin (Office of Government Commerce 2010b, 303).

Relaatio = Kahden ihmisen tai asian välinen yhteys tai vuorovaikutus. Konfiguraationhallinnassa relaatiolla tarkoitetaan kahden konfiguraatio-objektin välistä linkkiä, joka identifioi joko riippuvuussuhteen tai yhteyden niiden välillä. Esimerkiksi sovellukset voivat olla linkitettyjä palvelimiin joilla ne pyörivät. (Office of Government Commerce 2010e, 241.)

REST API = Rajapinta, minkä kautta voidaan käyttää funktioita HTTP-protokollaa hyödyntäen (IBM 2014f).

SDK = Software Developer Kit eli sovelluskehittäjän työkalut (IBM 2014e).

Service Asset = Mikä tahansa palveluntarjoajan resurssi tai kyky joka edistää palvelun tuottamista (Office of Government Commerce 2010e, 226).

TADDM = Tivoli Application Dependency Discovery Manager. IBM:n tekemä ohjelma, joka automatisoidusti etsii verkossa olevien sovellusten ja laitteiden kokoonpanotietoja ja niiden välisiä riippuvuustietoja eli relaatioita (IBM 2014m, 1).

1 JOHDANTO

Opinnäytetyössä tehdään integraatio ServiceNow-palvelunhallinta-alustan ja IBM:n Tivoli Application Dependency Discovery Manager -ohjelman välille. Opinnäytetyön aihe sai alkunsa Enfo Oyj:llä käynnissä olevasta ”Konfiguraation- ja muutoksenhallinnan työkalu -POC (TADDM)” -nimisestä projektista. Kyseisen projektin tarkoituksena on tuottaa Enfon sisäisen konfiguraatiohallinnan käyttöön prosessit ja työkalu, joiden avulla konfiguraatiotieto saataisiin kerättyä automaattisesti palvelimilta ja tuotua Enfon käyttämään ServiceNow-palvelunhallinta-alustaan asiantuntijoiden käyttöön. (Piippo 2014–10-7, 4.)

Kyseinen projekti on siinä vaiheessa, että konfiguraatiotietoja on kerätty palvelimilta TADDM-ohjelmaa käyttäen ja tietoja pitäisi saada siirrettyä ServiceNow-palvelunhallinta-alustaan asiantuntijoiden hyödynnettäväksi. Koska vielä ei kuitenkaan tiedetä, kuinka tietojen siirtäminen tulisi tehdä, selvitetään ja toteutetaan tietojen siirtäminen opinnäytetyössä. Tietojen siirtäminen eli integraatio toteutetaan siten, että TADDM-ohjelmasta tuodaan palvelimien väliset relaatiotiedot ServiceNow-alustaan. Näin saadaan luotua kanava, jota pitkin TADDM-ohjelmasta voidaan jatkossa siirtää mitä tahansa konfiguraatiotietoa ServiceNow-alustaan.

Enfo Oyj on pohjoismainen IT-palvelutalo, joka tarjoaa IT-ulkoistusta, talousprosessipalveluja sekä IT-konsultointipalveluja Suomessa, Ruotsissa, Norjassa ja Tanskassa. Enfo Oyj:llä on 50 vuoden kokemus tietotekniikkaratkaisujen ja -konseptien kehittämisestä ja yrityksessä työskentelee lähes 800 IT-ammattilaista. (Enfo Oyj 2014.)

ServiceNow on Enfo Oyj:n päivittäisessä palvelunhallinnassaan käyttämä palvelunhallinta-alusta. Alustan toimittaja on ServiceNow-niminen yritys, joka tarjoaa muille yrityksille pilvipalveluihin pohjautuvaa IT-toimintojen automatisointia (ServiceNow 2015d). TADDM on IBM:n tekemä ohjelma, joka automatisoidusti etsii verkossa olevien sovellusten ja laitteiden kokoonpanotietoja ja niiden välisiä riippuvuustietoja eli relaatioita (IBM 2014m, 1).

Koska ServiceNow on palvelunhallinta-alusta ja TADDM on konfiguraationhallintaohjelma, sisältää opinnäytetyö aluksi lyhyen kuvauksen palvelunhallinnasta ja konfiguraationhallinnasta. Seuraavaksi opinnäytetyössä perehdytään ServiceNow-alustaan, TADDM-ohjelmaan ja integraation toteutustapaan. Lopuksi opinnäytetyössä toteutetaan integraatio.

2 PALVELU JA PALVELUNHALLINTA

Palvelunhallinnan juuret ovat saaneet alkunsa perinteisistä palvelualoista, kuten lentoyhtiöistä, pankeista ja hotelleista (Office of Government Commerce 2007, 5). IT-palvelunhallinnan evoluutio sai puolestaan alkunsa kehittyvän teknologian alkaessa tukea jo olemassa olevia palveluja. Aluksi IT-ala oli keskittynyt pääasiassa sovelluskehitykseen, mutta uuden teknologian etujen hyödyntäminen tarkoitti sitä, että luoduista sovelluksista piti tehdä suurempia, liiketoimintoja tukevia palvelukokonaisuuksia. (Office of Government Commerce 2007, 3.)

2.1 ITIL-käsitteen syntyminen

Palvelunhallinnan käytäntöjen kasvaessa ja kehittyessä myös liiketoiminnan tarpeet kasvoivat. Jotta IT-palveluja käyttävien jatkuviin ongelmatilanteisiin pystyttiin vastaamaan, piti IT-palvelujen tähtäintä muuttaa. Näin saivat alkunsa IT-help deskit eli käyttäjien tukipalvelut. Samaan aikaan Iso-Britannian hallitus halusi tehostaa toimintaansa ja alkoi dokumentoimaan, kuinka parhaiten menestyvät yritykset ja organisaatiot hallitsivat tarjoamiaan palveluita. 1990-luvun alkuun mennessä Iso-Britannian hallitus oli tuottanut kirjasarjan, joka sisälsi yleiset käytännöt siitä, kuinka IT-palveluja hallitaan ja toteutetaan siten, että ne tukevat ja auttavat käyttäjiä. Tämä kirjasarja sai nimekseen IT Infrastructure Library, josta tulee lyhenne ITIL. (Office of Government Commerce 2007, 3.)

Kirjasarja kasvoi yli neljänkymmenen kirjan kokoelmaksi, ja se herätti suurta kiinnostusta Iso-Britanniassa IT-alan henkilöiden keskuudessa. Termistä IT-palvelunhallinta tuli yleisesti käytetty termi 1990-luvun puolivälin tienoilla ITIL-käytäntöjen suosion kasvaessa. ITIL:n seuraava, tarkistettu painos sai alkunsa 1990-luvun puolivälissä ja ITIL:n versio 2, kuten sitä laajalti nimitettiin, valmistui vuonna 2004. Versio 2 oli huomattavasti tiiviimpi paketti sisältäen vain yhdeksän kirjaa. Versio 2 keskittyi erityisesti kaventamaan teknologian ja liiketoiminnan välistä kuilua ja sen ohjeistukset keskittyivät ohjaamaan nimenomaan niitä prosesseja, joita tarvittiin palveluiden tehokkaaseen toimittamiseen asiakkaille. Samana vuonna kun versio 2 valmistui, OGC aloitti seuraavan suuren ITILiä koskevan uudistuksensa. IT-alan kantaviksi voimiksi alkoivat muodostua esimerkiksi virtualisointi ja ulkoistaminen, joten prosessipohjainen, ITILiin pohjautuva lähestymistapa täytyi jälleen uudistaa. Teknologia oli kehittynyt valtavasti ja palvelunhallinnassa eteen tuleviin haasteisiin piti kyetä vastaamaan. (Office of Government Commerce 2007, 3.)

ITIL-kirjasto koostuu nykyään kahdesta osasta: Ensimmäinen osa on viisiosainen ITIL Core -kirjasarja, joka tarjoaa parhaat käytännöt kaikentyyppisiä palveluja tarjoaville yrityksille. Toinen osa on ITIL Complementary Guidance -kirjasarja, joka täydentää Core-sarjaa antaen tarkempia ohjeita esimerkiksi eri teollisuuden sektoreille ja organisaatiotyypeille. (Office of Government Commerce 2010d, 7.)

Core-kirjasarjan kirjat ovat nimeltään Service Strategy, Service Design, Service Transition, Service Operation ja Continual Service Improvement (Office of Government Commerce 2010d, 7 - 8). ITIL Coren arkkitehtuuri perustuu palvelun elinkaarimalliin. Service Strategy voidaan ajatella akselina,

minkä ympäri elinkaari pyörii. Service Design, Service Transition ja Service Operation toteuttavat strategiaa ja Continual Service Improvement auttaa asettamaan ja priorisoimaan kehitysohjelmia ja -projekteja strategisten tavoitteiden pohjalta. (Office of Government Commerce 2010d, 24.) Tässä opinnäytetyössä tärkeimpänä tiedonlähteenä on teos Service Transition, jossa käsitellään konfiguraationhallintaa.

2.2 Palvelu

Office of Government Commerce määrittelee ITIL-kirjasarjassaan (2007, 5) palvelun ydinajatuksen vapaasti suomennettuna seuraavasti: Palvelu on keino - tai joukko keinoja, joiden avulla tuotetaan arvoa asiakkaalle. Asiakkaalla on jokin tavoite, johon hän haluaa päästä, ja palvelun avulla helpotetaan asiakkaan pääsyä tavoitteeseensa. Tavoitteeseen pääsyä helpotetaan poistamalla jokin tietty kustannus tai riski asiakkaalta. Asiakas siis ostaa palvelun, eikä hänen tarvitse itse huolehtia kyseiseen palveluun kuuluvien toimenpiteiden toteuttamisesta tai ylläpidosta. (Office of Governement Commerce 2007, 5.)

Jos asiakas esimerkiksi haluaa saada tallennustilaa verkkokauppansa tueksi, haluaa hän toisin sanoen saada tallennustilan hankintaan ja ylläpitoon liittyvän henkilökunnan, laitteet ja tuotantotilat eli koko siihen liittyvän infrastruktuurin. Asiakas ei kuitenkaan halua olla vastuussa kaikista edellä mainittuihin asioihin liittyvistä kuluista ja riskeistä, joten asiakas ostaa tallennustilan palveluna. Näin asiakkaan ei tarvitse itse vastata esimerkiksi tallennustilan toiminnasta tai ylläpidosta, vakuutuksista, turvamääräyksistä, ammattitaitoisesta henkilökunnasta, levytilan optimoinnista tai hetkellisistä käyttäjäkuormista. (Office of Governement Commerce 2010d, 16 - 17.)

2.3 Palvelunhallinta

Office of Government Commercen (2010d, 15) mukaan palvelunhallinta on joukko sellaisia organisaation toimintoja ja prosesseja, joiden avulla hallitaan palveluja niiden elinkaaren aikana. Toiminnoilla ja prosesseilla tarkoitetaan erityisesti palveluihin liittyvää strategiaa, palveluiden suunnittelua, siirtymiä, toiminnallisuutta ja palvelujen jatkuvaa kehittämistä. Palvelunhallinnan ydinasia on resurssien muuntaminen arvokkaiksi palveluiksi. Ilman palvelunhallintaa palveluja tarjoava organisaatio on vain joukko resursseja, jotka sellaisenaan tuottavat vain vähän arvoa asiakkaille. (Office of Governement Commerce 2010d, 15.)

Palvelunhallintaan vaikuttavat monet sellaiset haasteet, jotka erottavat palvelunhallinnan muista arvoa tuottavista toiminnoista. Esimerkiksi maanviljelijän tuotokset on helppo mitata, kun taas palvelunhallinnan tuotosten mittaaminen on vaikeaa. Maanviljelijä voi punnita ja näyttää tuottamansa viljamäärän, mutta koska palvelunhallinnan tulokset ovat aineettomia, on palveluntarjoajan vaikea punnita tai todistaa tuottamansa palvelun määrää. (Office of Government Commerce 2010d, 15.)

Menestyvillä palveluntarjoajilla on monia yhteisiä piirteitä ja toimintatapoja. Ne kaikki esimerkiksi tietävät, kuinka tuottaa arvoa asiakkailleen, ja ne ymmärtävät asiakkaan liiketoiminnan tavoitteet. Ne myös tietävät, mitä niiltä vaaditaan, jotta asiakas pääsee tavoitteisiinsa liiketoiminnassaan. Tämä ei johdu siitä, että ne pystyisivät reagoimaan asiakkaiden tarpeisiin niiden ilmetessä, vaan siitä, että tarpeet pystytään ennakoimaan esimerkiksi analysoimalla ja käyttäytymismalleja tutkimalla. Menestyvät palveluntarjoajat myös toteuttavat sellaisia palvelunhallinnan käytäntöjä, joiden avulla ne pystyvät nopeasti reagoimaan muutoksiin, jotka ovat yhtenäisiä ja joita voi jollain tapaa mitata. Ne pystyvät jatkuvasti analysoimaan ja hienosäätämään palvelutarjontaansa ja saavat näin ylläpidettyä vakaita, luotettavia ja silti muuntautumiskykyisiä palveluja, joiden avulla asiakas voi keskittyä omaan liiketoimintaansa. (Office of Government Commerce 2007, 4.)

ITIL on tarkoituksellisesti laadittu maalaisjärjellä ymmärrettäväksi lähestymistavaksi palvelunhallinnalle. ITIL:n mukaisessa toimintamallissa sovelletaan yleisiä IT-palveluja yhdistäviä käytäntöjä tavoitteena arvon tuottaminen liiketoiminnalle. ITIL tarjoaa siis puitteet ja ohjeet sellaisten palveluiden tuottamiseksi, jotka ovat tarkoitukseen sopivia, vakaita ja luotettavia. ITIL:n käytännöt edustavat luokkansa parhaiden palveluntarjoajien oppimisprosesseja ja johtajuusaatteita. Ne ovat ajan saatossa hyväksi koettuja ja testattuja, ja niitä voi soveltaa kaikentyyppisissä ja -kokoisissa palveluorganisaatioissa. (Office of Government Commerce 2007, 3 - 5.)

3 KONFIGURAATIONHALLINTA

Organisaatio voi olla tehokas vain, jos se hallitsee resurssejaan hyvin. Erityisen hyvin sen tulisi hallita niitä resursseja, jotka ovat elintärkeitä asiakkaan tai oman organisaation liiketoiminnalle. Näiden resurssien hallintaprosessia kutsutaan nimellä Service Asset and Configuration Management. (Office of Government Commerce 2010e, 65.)

Asset Management on prosessi, joka vastaa palvelun tuottamiseen liittyvistä taloudellisista resursseista koko palvelun elinkaaren ajan (Office of Government Commerce 2010e, 226). Se tarjoaa tiedon siitä, mitä resursseja palvelujen tuottamiseen on käytettävissä, kuinka arvokkaita resurssit ovat ja kuka resursseista vastaa. (Office of Government Commerce 2010e, 65; Office of Government Commerce 2010e, 226.)

Configuration Management puolestaan on prosessi, minkä avulla ylläpidetään konfiguraatio-objektien tietoja koko niiden elinkaaren ajan (Office of Government Commerce 2010e, 230). Service Asset and Configuration Management on siis osa palvelunhallintaa ja tarkoittaa vapaasti suomennettuna palvelun tuottamiseen tarvittavien resurssien ja konfiguraatio-objektien hallintaprosessia. (Office of Government Commerce 2010e, 65.)

3.1 Konfiguraatio-objektit

Konfiguraatio-objekti on jokin resurssi, palvelukomponentti tai jokin muu asia tai esine, jota hallitaan konfiguraationhallinnan avulla. Konfiguraatio-objektit voivat vaihdella suurestikin esimerkiksi monimutkaisuutensa, kokonsa ja tyyppinsä puolesta. Kokonainen palvelu laitteineen, ohjelmistoineen, dokumentteineen ja tukihenkilöstöineen voi muodostaa yhden objektin, mutta myös jokin hyvin pieni laitteen osa tai ohjelmamoduuli voi olla yksi objekti. (Office of Government Commerce 2010e, 67.)

Konfiguraatio-objekteja voidaan koota ryhmiksi ja niitä voidaan hallita ryhmänä. Ne voidaan jaotella esimerkiksi seuraaviin ryhmiin: palvelun elinkaari-, palvelu- ja organisaatio-objektit sekä sisäiset ja ulkoiset objektit. (Office of Government Commerce 2010e, 67 - 68.)

Palvelun elinkaariobjektit sisältävät esimerkiksi palvelunhallinnan suunnitelmat sekä julkaisu-, muutos- ja testisuunnitelmat. Ne tarjoavat kuvan palveluntarjoajan palveluista ja siitä, kuinka näitä palveluja toimitetaan. Ne tarjoavat myös kuvan siitä, millaisia etuja palveluiden myötä on odotettavissa ja milloin ne toteutuvat. (Office of Government Commerce 2010e, 67.)

Palveluobjekteihin kuuluvat esimerkiksi palvelujen toteutukseen ja resursseihin liittyvät objektit, kuten tilat, pääomat ja ihmiset sekä palvelumallit ja palvelupaketit. Organisaatio-objekteihin puolestaan kuuluvat esimerkiksi dokumentit organisaation liiketoimintastrategiasta tai muista käytännöistä, jotka koskevat yritystä. (Office of Government Commerce 2010e, 67 - 68.)

Sisäiset objektit sisältävät ne aineettomat (esim. ohjelmat) ja aineelliset (esim. datakeskus) komponentit, jotka tarvitaan yrityksen tarjoamien palvelujen tuottamiseen ja ylläpitoon. Ulkoiset objektit sisältävät esimerkiksi asiakkaiden kanssa tehdyt sopimukset ja ulkopuolisten palveluntoimittajien julkaisut. (Office of Government Commerce 2010e, 67 - 68.)

3.2 Konfiguraationhallinnan tarkoitus

Hajautetuissa ympäristöissä yksittäiset objektit kuuluvat useisiin eri palveluihin ja konfiguraatiorakenteisiin. Esimerkiksi työntekijä voi käyttää tietokonetta, joka on rakennuksen verkossa, mutta tietokoneella voi olla ohjelma, joka on yhteydessä maailman toisella puolella sijaitsevaan tietokantaan. Tällöin muutos joko rakennuksen verkkoon, tietokoneelle asennettuun ohjelmaan tai kaukana sijaitsevaan tietokantaan voi vaikuttaa työntekijän lisäksi hänen koko liiketoimintaprosessiinsa. (Office of Government Commerce 2010e, 66.) Suurien ja monimutkaisten IT-palveluiden ja infrastruktuurien hallitsemisen avuksi tarvitaan tukijärjestelmää, jota kutsutaan nimellä Configuration Management System eli vapaasti suomennettuna konfiguraationhallintajärjestelmä. (Office of Government Commerce 2010e, 68 - 69.)

CMS pitää sisällään jokaisen konfiguraatio-objektin tiedot. Esimerkiksi yksi palveluobjekti voi sisältää tiedon palvelun tarjoajasta, hinnasta, ostopäivästä, lisenssien ja huoltosopimusten uusintapäivästä sekä siihen liittyvistä sopimuksista. CMS ylläpitää myös kaikkien palveluobjektien välisiä relaatiotietoja sekä niihin liittyviä ongelmia, häiriöitä, tunnettuja virheitä ja muutoksia. CMS voi myös sisältää tiedot esimerkiksi työntekijöistä, tavarantoimittajista, sijainneista, liiketoimintayksiköistä, asiakkaista ja käyttäjistä. Datatasolla CMS sisältää yhden tai useamman tietokannan, joita kutsutaan nimellä CMDB. CMS siis ikään kuin kokoaa CMDB:t yhteen ja tarjoaa yhden reitin kaikkiin näihin tietokantoihin. (Office of Government Commerce 2010e, 68 - 69.)

Konfiguraationhallinnan tavoitteena on tukea palvelunhallintaa tarjoamalla tarkkoja konfiguraatiotietoja, joiden avulla ihmiset voivat tehdä oikea-aikaisia päätöksiä ja ratkaista esimerkiksi häiriöilmoituksia ja ongelmia nopeammin. Tavoitteena on myös minimoida huonosti konfiguroiduista palveluista johtuvat ongelmat ja optimoida resursseja. (Office of Government Commerce 2010e, 65.)

Konfiguraationhallinta varmistaa, että ne komponentit, joista kokonaiset palvelut, järjestelmät tai tuotteet rakentuvat, ovat hyvin ylläpidettyjä. Lisäksi se varmistaa, että niihin kohdistuvat muutokset ovat hallittuja. Konfiguraationhallinta myös tarjoaa kuvan siitä, miten palvelut, resurssit ja koko infrastruktuuri rakentuu tallentamalla tiedon objektien välisistä relaatioista eli tiedon siitä, kuinka ne liittyvät toisiinsa. (Office of Government Commerce 2010e, 65.)

Konfiguraationhallinta tuottaa siis loogisen kuvan infrastruktuurin rakenteesta. Kuvan avulla muut liiketoiminnan prosessit pääsevät käsiksi arvokkaaseen tietoon, ja voivat esimerkiksi tutkia häiriöiden ja ongelmien vaikutuksia ja syitä sekä ehdotettujen muutosten vaikutuksia. Kokonaiskuvaa voidaan hyödyntää myös suunniteltaessa teknologia- ja ohjelmistopäivityksiä, uusia palveluja tai olemassa olevien palvelujen muutoksia. (Office of Government Commerce 2010e, 66 - 67.)

Koska konfiguraatiotieto kehittyy palvelujen kehittyessä, on tietojen keräämiseen ja ylläpitoon suositeltavaa kehittää automatisoituja prosesseja. Näin voidaan pienentää kuluja ja vähentää virheiden mahdollisuuksia. Erilaiset tietoja automaattisesti etsivät ja keräävät työkalut, inventointi- ja auditointityökalut ja verkonhallintatyökalut voidaan liittää CMS:ään. Työkalujen avulla CMS:ään voidaan ensimmäisen kerran tuoda dataa ja myöhemmin vertailla todellisen tilanteen ja tietokannassa olevien tietojen paikkansapitävyyttä toisiinsa nähden. (Office of Government Commerce 2010e, 69.) IBM:n TADDM-ohjelma on nimenomaan konfiguraatiotiedon automatisoituun keräämiseen kehitetty työkalu (IBM 2014m, 1).

4 SERVICENOW

ServiceNow on pilvipalveluihin perustuva alusta, minkä päällä toimii joukko IT-palveluiden automatisointi- ja hallintatuotteita. Kaikilla tuotteilla on näin ollen yksi yhteinen käyttöliittymä, koodipohja ja tietomalli. ServiceNow:n tarkoituksena on luoda tiedoille yksi järjestelmä, jota kaikki organisaation työntekijät käyttävät esimerkiksi tiedonlähteenä ja minkä avulla he raportoivat tekemäänsä työtä. (ServiceNow 2015d, 3 - 6.) Näin esimerkiksi asiantuntijoiden ei tarvitse etsiä ja koota tarvitsemaansa tietoa useasta eri lähteestä ja johdolla on suora pääsy ajantasaiseen tietoon yrityksen toiminnasta (ServiceNow 2015f).

ServiceNow:n avulla organisaatio voi myös automatisoida IT-palveluihin kuuluvia manuaalisia tehtäviä, luoda omiin tarpeisiin sopivia sovelluksia, parantaa IT prosessien tehokkuutta ja raportoida tehokkuutta sekä kustannuksia. Organisaatio voi myös pienentää infrastruktuurin ylläpidon vaatimuksia, sillä ServiceNow:n tuotteita käytetään internetselaimella ilman erillistä sovellusta. (ServiceNow 2015f.) ServiceNow-alustan yksi osa-alue on konfiguraationhallinta ja se sisältää CMDB:n (ServiceNow 2015e).

ServiceNow-alusta oli opinnäytetyötä tehtäessä Enfo Oyj:n käytössä palvelunhallinta- ja työnohjausjärjestelmänä. Opinnäytetyössä täydennettiin ServiceNow-alustan CMDB:n sisältöä tuomalla sinne TADDM-ohjelman keräämää konfiguraatiotietoa.

5 TADDM

TADDM on IBM:n tekemä ohjelma, joka automatisoidusti etsii verkossa olevien sovellusten ja laitteiden kokoonpanotietoja ja niiden välisiä riippuvuustietoja eli relaatioita. TADDM kykenee löytämään verkkoympäristössä esimerkiksi seuraavana lueteltujen komponenttien kokoonpano- ja relaatiotiedot:

- sovelluskomponentit, kuten web-palvelimet, sovelluspalvelimet ja tietokannat
- järjestelmäkomponentit, kuten käyttöjärjestelmät, kuormanjakajat ja tietokantapalvelimet
- verkkokomponentit, kuten reitittimet, kytkimet ja palomuurit
- erilaiset palvelut, kuten Domain Name Service, Lightweight Directory Access Protocol ja Network File System.

TADDM pystyy myös löytämään yksittäisten ohjelmaprosessien väliset riippuvuudet ja tiedon siitä, toimivatko ne esimerkiksi Microsoft Windows- vai Linux-ympäristöissä. (IBM 2014m, 1 - 6.)

5.1 TADDM:n käyttötarkoitus

TADDM tuottaa siis tiedon liiketoiminnan sovellusten, ohjelmien ja fyysisten komponenttien välisistä suhteista ja riippuvuuksista. Tämän tiedon avulla on esimerkiksi helpompi nähdä millainen vaikutus johonkin yksittäiseen komponenttiin kohdistuvalla muutoksella on laajemmalla tasolla. (IBM 2014m, 1 - 2.) Näin organisaatio pystyy tehokkaammin suunnittelemaan esimerkiksi konfiguraatiomuutoksia ja ymmärtämään paremmin komponenttitason tapahtumien vaikutuksia. (IBM 2014m, 2 - 3.)

TADDM tarjoaa suorittavalle henkilöstölle myös kokonaiskuvan sovelluksista, jolloin henkilöstö saa nopeasti käsityksen sovellusten rakenteesta, konfiguraatiosta ja muutoshistoriasta. Kuvan avulla voidaan esimerkiksi nopeasti eristää konfiguraatioista johtuvia sovellusten ongelmia ja näin vähentää vianselvitykseen kuluvaa aikaa huomattavasti. (IBM 2014m, 2 - 3.)

5.2 TADDM:n toimintaperiaate

TADDM:n toimintaprosessia ohjaa agentiton etsintämoottori eli TADDM-palvelimelle asennettu ohjelma. Agentiton tarkoittaa sitä, että etsintämoottori lähettää tutkittavaan kohteeseen sensoreja, eikä kohteeseen siksi tarvitse asentaa mitään. Sensoreiden etuna on nimenomaan se, ettei etsintäprosessi ole riippuvainen siitä onko tutkittavaan kohteeseen asennettu agentti vai ei. Sensorit kuluttavat lisäksi hyvin vähän verkon kaistaa ja tutkittavan kohteen prosessoriresursseja (alle 1 %:n aktiivisena). (IBM 2014m, 33.)

Ennen etsintäprosessin aloittamista TADDM:n etsintämoottorille tulee syöttää kolme tietoa. Ensimmäisenä etsintämoottori tarvitsee tiedon etsinnän laajuudesta. Tämä on tyypillisesti jokin IP-osoiteavaruus, aliverkko tai tietty osoite. Seuraavaksi etsintämoottori tarvitsee listan niistä tunnuksista joita se tarvitsee ottaessaan yhteyden ja kirjautuessaan tutkittavaan kohteeseen tietojen luke-

mista varten. Yhteysmekanismi vaihtelee tutkittavan kohteen mukaan. Etsintämoottori voi käyttää esimerkiksi SSH-yhteyttä Unix-pohjaisten käyttöjärjestelmien kohdalla ja WMI-yhteyttä tutkiessaan Windows-käyttöjärjestelmää. Lopuksi TADDM:n etsintämoottori tarvitsee tiedon siitä koska sen tulee aloittaa etsintäprosessi. Prosessin voi määrittää käynnistymään vain manuaalisesti, ajastettuna tai jonkin ulkoisesti annettavan eventin käynnistämänä. (IBM 2014m, 33 - 34.)

Kun etsintäprosessi käynnistyy, etsintämoottori käyttää standardiprotokollia tunnistaakseen jokaisen määritellyssä osoitelaajuudessa sijaitsevan laitteen. Seuraavaksi TADDM lähettää jokaiseen validiin IP-osoitteeseen etsintäsensorin, joka löytää ja kategorisoi tutkittavan kohteen. (IBM 2014m, 34.) Käyttäjää emuloiva sensori kirjautuu sisään kohdejärjestelmään ja suorittaa käyttöjärjestelmälle natiiveja komentoja kerätäkseen kohteesta sen konfiguraatio- ja relaatiotiedot (IBM 2014m, 8 - 9; IBM 2014m, 34).

Tutkimusprosessi on iteratiivinen, eli jokainen sensori voi laukaista oman alisensorinsa (IBM 2014m, 34). Sensori voi esimerkiksi ensin tutkia onko kohde verkkolaite vai tietokonejärjestelmä. Kun etsintämoottori saa kohteesta lisätietoja, lähetetään kohteeseen tarkempia ja tarkempia sensoreita joiden avulla kohdetta tutkitaan. (IBM 2014m, 8 - 9.) Jos sensori esimerkiksi päättelee, että isäntäkone on tietokonejärjestelmä, voi se laukaista sensorin, joka tutkii koneelle asennetut sovellukset ja palvelut. Iteratiivinen prosessi jatkuu niin kauan kunnes kohteessa ei ole enää tutkittavaa. (IBM 2014m, 34.) Sensoreiden tehtävä on siis löytää konfiguraatio-objektit, luoda niistä TADDM-palvelimen tietokantaan sopiva objekti ja lähettää tieto palvelimelle. Etsintä on monisäikeinen prosessi ja sitä tapahtuu yleensä useammassa kohteessa samanaikaisesti. (IBM 2014m, 8 - 9).

Etsintäprosessin lopuksi TADDM käsittelee löydettyjen objektien tiedot, tallentaa ne omaan CMDB:een ja luo topologisen kuvan infrastruktuurista (IBM 2014m, 35). Tämän TADDM tekee analysoimalla ja mahdollisesti yhdistelemällä kerättyä tietoa jo olemassa olevaan tietoon (IBM 2014m, 8 - 9). Kun TADDM:n etsintäprosessi seuraavan kerran suoritetaan, TADDM päivittää muuttuneet konfiguraatio- ja relaatiotiedot tietokantaansa säilyttäen samalla täydellisen muutoshistorian tiedoista (IBM 2014m, 35). TADDM-ohjelman konsoli tarjoaa analysointi- ja raportointinäkymän koko TADDM-palvelimen CMDB:n sisältämään tietoon (IBM 2014m, 8 - 9).

5.3 Common Data Model

TADDM tallentaa tietonsa sisäisesti käyttäen Java-objektien hierarkkista mallia, jota kutsutaan lyhenteellä CDM. CDM pohjautuu relaatiotietokantaan ja se koostuu objekteista, jotka edustavat löydettyjä elementtejä. CDM sisältää siis löydetyt objektit (esimerkiksi tietokonejärjestelmät tai tietokannat) sekä objekteihin liittyvät tarkemmat tiedot (esimerkiksi käyttöjärjestelmät ja konfiguraatioarvot), jotka esitetään objekteihin sisältyvinä omina objekteinaan. (IBM 2014c.)

CDM sisältää useita erityyppisiä objekteja. Luokaksi kutsutaan objektia, jota käytetään kokoamaan toisiinsa liittyvät attribuutit yhteen (esim. palvelimen tiedot). Luokat sisältävät attribuutteja, ne implementoivat rajapintoja ja voivat olla osallisena relaatioissa. Luokat ovat hierarkkisia, mikä tarkoit-

taa sitä, että ne perivät emoluokan ominaisuudet. Attribuutti määrittelee jonkin ominaisuuden joka objektilla on. Attribuutit ovat tavallaan adjektiiveja, jotka kuvailevat objekteja ja jotka erottavat saman luokan objektit toisistaan. Esimerkiksi valmistajatieto on attribuutti. (IBM 2014g.)

Rajapinta sallii attribuuttien uudelleenkäyttämisen erilaisten objektityyppien kesken. Esimerkiksi VersionString–attribuutti on usealla eri luokkaan kuuluvalla objektilla. Sen sijaan että attribuutti kopioitaisiin jokaiselle objektille ja luokalle erikseen, luodaan rajapinta joka edustaa tätä attribuuttia. Näin mikä tahansa luokka joka implementoi tätä rajapintaa, saa käyttöönsä rajapinnan määrittämän attribuutin aivan kuin se kuuluisi kyseiseen luokkaan. (IBM 2014g.)

Relaatio kertoo millainen yhteys kahden objektin välillä on. Esimerkiksi ”installedOn” – tyyppinen relaatio kertoo, että lähdeobjekti on asennettu kohdeobjektiin (esimerkiksi ohjelma on asennettu palvelimelle). Kahden objektin välillä voi olla yksi tai useampi relaatio ja jokainen relaatio assosioi kaksi objektia toisiinsa. (IBM 2014g.)

CDM tarjoaa johdonmukaisen määritelmän erityyppisille objekteille ja parhaiksi havaitut käytännöt TADDM:n tietokantaan tallennettavalle sisällölle. Se standardisoi esimerkiksi tallennettavien tietojen nimeämiskäytännöt, tietotyypit, attribuutit ja luokat siten, että kaikki ohjelmat, jotka käyttävät samaa mallia, voivat helposti vaihtaa tietoja keskenään. CDM:ään sisältyy lukuisia standardeja, mukaan lukien ITIL–käytännöt. (IBM 2014g.)

5.4 SDK ja rajapinnat

TADDM ohjelman asennuspaketti sisältää sovelluskehittäjän työkalut, joiden avulla pääsee käsiksi TADDM:n tietokannan tietoihin. Tietoja käsitellään ja niitä luetaan tietokannasta erityisten rajapintojen eli APIen avulla. TADDM:n työkalut sisälsivät seuraavat rajapinnat: Java API, SOAP API, REST API ja Command-line interface API. Näiden rajapintojen avulla TADDM:n tietokannasta voidaan siis esimerkiksi lukea löydettyjen sovellusten tiedot, komponenttitiedot ja relaatiotiedot. Rajapintojen avulla voidaan myös hallita TADDM:n toimintaa, mutta tässä opinnäytetyössä hyödynnettiin vain rajapintojen tarjoamaa tietojen lukuominaisuutta. (IBM 2014e.)

6 INTEGRAATIO

Integraatiossa TADDM-ohjelmasta siirrettiin palvelinten välisiä relaatiotietoja ServiceNow-alustaan. Relaatiotiedot rajattiin koskemaan pelkästään palvelimia, jotta siirrettävän informaation määrä pysyi kohtuullisena. Integraation tietolähteenä oli IBM:n Knowledge Center sekä TADDM:n asennuspaketin mukana tuleva dokumentaatio. Enfo Oyj on toteuttanut useita integraatioita esimerkiksi IBM:n Tivoli-ohjelmien ja ServiceNow-alustan välille, joten Enfo Oyj:n asiantuntijoiden apua käytettiin myös tarvittaessa hyödyksi.

Enfo Oyj:llä oli käytettävissään kaksi palvelinta, joihin TADDM-ohjelma oli asennettu. Toisen palvelimen TADDM-ohjelma sisälsi vain keinotekoisesti tuotettua testidataa, kun taas toiselle palvelimelle asennettu ohjelma sisälsi aitoa Enfo Oyj:n keräämää dataa.

6.1 Integraation arkkitehtuuri ja rajapinnan valitseminen

Enfo Oyj:n asiantuntija suositteli toteuttamaan integraation seuraavasti:

- Palvelimelle A asennettu Mule-ohjelma kysyy ajastetusti tarvittavat tiedot TADDM-palvelimelta B.
- TADDM-palvelin B palauttaa tiedot JSON-muodossa palvelimelle A.
- Palvelin A muotoilee tietoja ja lähettää ne välitietokantaan palvelimelle C.
- ServiceNow-alusta kutsuu palvelinta A, joka kysyy tiedot välitietokannasta palvelimelta C ja palauttaa ne ServiceNow-alustaan.
- ServiceNow-alusta tekee tietojen perusteella muutokset tietokantaansa. (Savolainen 2014.)

Arkkitehtuuri muodostettiin edellä mainitulla tavalla, koska TADDM-palvelimelle B ei haluttu tietoturvan vuoksi sallia suoraa pääsyä ServiceNow-alustasta. Aiempien integraatioiden vuoksi ServiceNow-alustan pääsy palvelimelle A oli jo sallittu, joten senkään vuoksi ei tarvittu tehdä muutoksia. Mikäli tulevaisuudessa ServiceNow-alusta vaihtuisi toiseen, ei koko integraatioketju myöskään pääsisi katkeamaan tällaisessa toteutuksessa. Jos oletetaan, että esimerkiksi kymmenen eri tietolähdettä olisi integroitu suoraan ServiceNow-alustaan, pitäisi alustan muuttuessa integraatio tehdä jokaiseen kymmeneen tietolähteeseen uudelleen. Jos integraatiot on toteutettu välitietokantaa hyödyntäen, tarvitsisi alustan vaihtuessa tehdä vain yksi uusi integraatiokanava. (Savolainen 2014.)

Integraatio aloitettiin selvittämällä, mikä luvussa 4 mainituista rajapinnoista olisi sopivin TADDM:n tietokannan tietojen lukemiseen. Toteutustavaksi valittiin REST API, koska REST APIa käyttämällä voidaan kehittää sovelluksia millä tahansa ohjelmointikielellä, joka tukee HTTP-kutsuja (IBM 2014f).

Enfo Oyj oli aiemmin toteuttanut integraatioita HTTP-protokollaa hyödyntäen, mikä myös tuki REST API:n valintaa. Palvelimelle A asennettu Mule-ohjelma pystyi tekemään HTTP-kutsun, ja se sisälsi valmiin toteutuksen, jota hyödyntämällä haluttu tieto saatiin tallennettua välitietokantaan C-palvelimelle. ServiceNow-alusta sisälsi lisäksi valmiin koodikirjaston, minkä avulla tieto saatiin haettua välitietokannasta C-palvelimelta. (Savolainen 2014.) Opinnäytetyössä tehtäväksi jäi siis

- MQL kyselylauseen kirjoittaminen (ks. seuraava kappale)
- TADDM-palvelimelta kyselyn vastauksena saatavan JSON-tekstin muotoilu ja lähetys välitietokantaan
- tietokannan taulun tekeminen välitietokantaan
- triggereiden ja stored proceduren tekeminen välitietokannan tauluun
- tietojen hakeminen välitietokannasta ServiceNow-alustaan
- tietojen tallennus ja relaatioiden käsittely ServiceNow-alustassa.

6.2 Model Query Language

REST API tarjosi kaksi vaihtoehtoa tietojen kyselyyn TADDM-palvelimelta. Ensimmäinen vaihtoehto oli model object class resource, minkä esimerkkikyselyn osoite näyttää seuraavalta:

```
http://example.com:9430/rest/model/ComputerSystem?depth=2
&feed=xml&OSRunning.OSName=Linux&position=5
```

Tämä kysely näyttää tiedot siitä tietokannassa järjestyksessä viidentenä olevasta tietokonejärjestelmäobjektista, minkä OSRunning-attribuutti on ”Linux” (IBM 2014k). Toinen vaihtoehto oli MQL, minkä esimerkkikyselyn osoite näyttää seuraavalta:

```
http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,
OSRunning.OSName%20from%20ComputerSystem&position=2&
fetchSize=2&feed=xml&depth=2&position=4
```

Tämä esimerkki tekee ”select displayName,OSRunning.OSName from ComputerSystem” -kyselyn TADDM:n tietokantaan. Näistä kahdesta vaihtoehdosta valittiin jälkimmäinen, MQL, koska se oli joustavampi kuin ensimmäinen vaihtoehto. (IBM 2014k.) MQL-kysely on purettu osiin ja parametrien tarkoitus on selitetty tarkemmin erillisessä liitteessä (Liite 1) (IBM 2014j). Kyseinen liite on kirjoitettu englanniksi Enfo Oyj:n pyynnöstä.

IBM:n Knowledge Center (IBM 2014d) tarjosi esimerkin siitä, kuinka MQL-kyselyn pystyi tekemään helposti internetselaimella ja näin kokeilemaan kyselyn toimivuutta. Knowledge Center ei kuitenkaan antanut valmista esimerkkiä siitä, miltä kyselyn tulos näyttäisi selaimessa, joten tätä kokeiltiin ensin Enfo Oyj:n omistamaan TADDM:n testi-instanssiin. Esimerkkiosoitetta muokattiin yksinkertaisemmaksi IBM:n Knowledge Centerin ohjeiden mukaisesti (IBM 2014i; IBM 2014h), jolloin saatiin seuraavanlainen osoite (palvelimen tarkkaa osoitetta ei voida julkaista salassapitovelvollisuuden vuoksi):

```
http://TADDM_palvelimen_osoite:9430/rest/model/MQLQuery?query=select%20*%20
from%20ComputerSystem&feed=json&fetchSize=1&position=1
```

Navigoimalla internetselaimella edellä mainittuun osoitteeseen saatiin selainikkunaan näkyviin seuraava JSON-objekti:


```
{
    "lastModifiedTime": 1420279894999,
    "oslcChangeRegistrationTime": 1420279894999,
    "oslcRegistrationTime": 1420279894999,
    "isPlaceholder": false,
    "OSInstalled": [
        {
            "guid": "F91DB3B2068030FD93484A7934888C43",
            "_class": "Linux"
        }
    ],
    "numCPUs": 2,
    "guid": "225B1B2BCF3633649D88351C1D148340",
    "OSRunning": {
        "guid": "F91DB3B2068030FD93484A7934888C43",
        "_class": "Linux"
    },
    "_class": "LinuxUnitaryComputerSystem",
    "createdBy": "f255760",
    "lastModifiedBy": "f255760",
    "bidiFlag": 3,
    "displayName": "JsonRestExample2",
    "signature": "JsonRestExample2"
}
```


6.3 NetworkConnection-luokka

Seuraavaksi selvitettiin, mitä tietoja TADDM:n tietokannasta tarvittaisiin. IBM:n Knowledge Centerin (IBM 2014l) mukaan TADDM:n asennuspaketti sisälsi HTML-muotoisen dokumentaation TADDM:n tietokannan rakenteesta. Tämän dokumentaation avulla opinnäytetyössä päädyttiin käyttämään NetworkConnection-luokan tietoja. Dokumentaation mukaan NetworkConnection-luokka edustaa abstraktia verkkoyhteyttä kahden ComputerSystem-luokan objektin välillä. Luokka tarjoaa tietoa näistä kahdesta tietokonejärjestelmästä. (IBM 2014a.)

NetworkConnection-luokassa oltiin kiinnostuneita erityisesti kahdesta attribuutista. Nämä olivat FromComputerSystem ja ToComputerSystem. FromComputerSystem kertoo, mistä ComputerSystem-luokan objektista verkkoyhteys saa alkunsa, ja ToComputerSystem kertoo, mihin ComputerSystem-luokan objektiin verkkoyhteys päättyy. (IBM 2014a.) Koska kahden palvelinobjektin välisen relaation tekemiseen ServiceNow-alustassa tarvittiin sekä From- että ToComputerSystem-objektien displayName- ja locationTag-attribuutit, eivät kyseiset attribuutit saaneet olla tyhjiä. Niinpä MQL-kyselylause muotoutui seuraavaksi:

```
http://TADDM_palvelimen_osoite:9430/rest/model/MQLQuery?
query=SELECT%20*%20FROM%20NetworkConnection%20
WHERE%20fromComputerSystem.locationTag%20is-not-null%20and%20
fromComputerSystem.displayName%20is-not-null%20and%20
toComputerSystem.locationTag%20is-not-null%20and%20
toComputerSystem.displayName%20is-not-null&feed=json&depth=2&fetchSize=10000
```

Edellä esitetty kysely hakee NetworkConnection-luokasta sellaiset rivit, joissa

- fromComputerSystem.locationTag ei ole tyhjä
- fromComputerSystem.displayName ei ole tyhjä
- toComputerSystem.locationTag ei ole tyhjä
- toComputerSystem.displayName ei ole tyhjä.

6.4 Taulun teko välitietokantaan

Koska Enfo Oyj oli toteuttanut aiemmin samankaltaisia integraatioita siten, että tiedot tallennetaan välitietokantaan ennen ServiceNow-alustaan lataamista, tuli myös TADDM-palvelimelta saatavat tiedot tallentaa välitietokantaan (Savolainen 2014). Tietokantaan luotiin uusi taulu seuraavalla luontilauseella:

```
CREATE TABLE [dbo].[TADDMData](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [ModelObjectGuid] [nvarchar](40) NOT NULL,
    [FromGuid] [nvarchar](40) NOT NULL,
    [FromAssetTag] [nvarchar](255) NOT NULL,
    [FromLocationTag] [nvarchar](20) NOT NULL,
    [FromDisplayName] [nvarchar](255) NOT NULL,
    [ToGuid] [nvarchar](40) NOT NULL,
    [ToAssetTag] [nvarchar](255) NOT NULL,
    [ToLocationTag] [nvarchar](20) NOT NULL,
    [ToDisplayName] [nvarchar](255) NOT NULL,
    [Created] [datetime] NULL,
    [Updated] [datetime] NULL,
    [Active] [tinyint] NULL,
    [ForcedUpdate] [datetime] NULL,
 CONSTRAINT [PK_TADDMData] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IG-
NORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
```

Koska tietokantaan haluttiin – mahdollista myöhemmin tapahtuvaa virheiden selvitystyötä helpottamaan – tallentaa tieto uuden rivin luontihetkestä sekä rivin muokkaushetkestä, tehtiin uuteen tauluun kaksi triggeriä. Ensimmäinen trigger päivittää Created-attribuutin arvoa automaattisesti aina, kun uusi rivi syntyy tauluun, ja toinen päivittää Updated-attribuutin arvoa aina rivin päivittyessä. Created-attribuuttia päivittävä Trigger luotiin seuraavalla luontilauseella:

```
CREATE TRIGGER [dbo].[TADDM_after_insert]
ON [dbo].[TADDMData]
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;
    UPDATE dbo.TADDMData
        SET TADDMData.Created = CURRENT_TIMESTAMP
            FROM inserted i
                INNER JOIN dbo.TADDMData taddm
                    ON i.id = taddm.id;
END
```

Updated-attribuuttia päivittävä Trigger puolestaan luotiin seuraavalla luontilauseella:

```
CREATE TRIGGER [dbo].[TADDM_after_update]
ON [dbo].[TADDMData]
AFTER UPDATE
AS
BEGIN
SET NOCOUNT ON;
    UPDATE dbo.TADDMData
        SET TADDMData.Updated = CURRENT_TIMESTAMP
            FROM inserted i
                INNER JOIN dbo.TADDMData taddm
                    ON i.id = taddm.id;
END
```

Lopuksi uuteen tauluun tehtiin stored procedure. Kyseinen procedure oli tarkoitus ajaa aina heti sen jälkeen, kun data oli tallennettu tietokantaan. Sen tarkoituksena oli päivittää uudesta taulusta niiden rivien Active-attribuutti nollaksi, jotka eivät saaneet päivitystä kyseisellä ajokerralla. Ajatuksena oli, että jostain on pystyttävä tunnistamaan, minkä rivin tieto on edelleen pätevä ja minkä ei. Stored procedure luotiin seuraavalla luontilauseella:

```
CREATE PROCEDURE [dbo].[DeactivateTADDMData]
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE tietokannan_nimi.dbo.TADDMData
        SET Active = 0
            WHERE ForcedUpdate <
                (SELECT MAX(ForcedUpdate)
                    FROM tietokannan_nimi.dbo.TADDMData)
END
```

Stored proceduren toimintaperiaate on seuraava: Oletetaan, että TADDM:n tietokannasta haetaan tiedot ensimmäisen kerran 24.1.2015 klo 12.00 ja tietokannasta löytyvät tietueet A ja B. Tällöin välitietokantaan tallennetaan rivit A ja B ja ne molemmat saavat ForcedUpdate-attribuutin arvoksi aikaleiman 2015-01-24 12:00:00.000 ja Active-attribuutin arvoksi 1. Kun stored procedure käynnistyy rivien tallentamisen jälkeen, ei se muuta kummankaan rivin Active-attribuutin arvoa, koska molemmilla riveillä on sama aikaleima. Oletetaan seuraavaksi, että TADDM:n tietokannasta haetaan tiedot seuraavan kerran 25.1.2015 klo 12.00 ja tietue A ei enää olekaan tietokannassa. Tällöin välitietokantaan viedään ainoastaan tietue B, minkä ForcedUpdate-attribuutin arvoksi päivittyy uusi aikaleima 2015-01-25 12:00:00.000. Tietueen A aikaleima säilyy muuttumattomana. Kun stored procedure nyt käynnistyy, se asettaa tietueen A Active-attribuutin arvoksi 0, koska ForcedUpdate-attribuutin aikaleima on vanhempi kuin uusin olemassa oleva aikaleima. Näin voidaan tunnistaa poistuneet tietueet sekä samalla nähdä, koska kyseiset poistuneet tietueet ovat vielä olleet voimassa.

6.5 MQL-kyselyn tekeminen, tulosten käsittely ja tallennus välitietokantaan

Enfo Oyj:n omistamalle palvelimelle oli asennettu Mule-ohjelma, jota käytettiin MQL-kyselyn lähettämiseen TADDM-palvelimelle. MQL-kyselyn suorittamiseksi Mulen konfiguraatiotiedostoon tuli lisätä seuraavat rivit:

```
<process name="TADDM_Data">
    <inbound-endpoint
        proto="http"
        method="GET"
        address=""
        httpConnectTimeout="600000"
        httpReadTimeout="600000">
        <header
            name="Authorization"
            value="" />
    </inbound-endpoint>
    <endpoint
        proto="javascript"
        address="./js/TADDM.js"
        payload="inbound-endpoint" />
    <endpoint
        proto="SQL"
        address="jdbc:sqlserver::;DatabaseName="
        payload="previous">
      <header
        name="username" value="" />
      <header
        name="password" value="" />
    </endpoint>
</process>
```

Address-attribuutin arvoksi annettiin luvussa 6.3 esitetty MQL-kyselylause eli HTTP-osoite, johon Mule-ohjelma tekee HTTP GET -kutsun (Savolainen 2014). Edellä esitetystä konfiguraatiotiedoston esimerkistä on poistettu esimerkiksi TADDM-ohjelman vaatima käyttäjätunnus ja salasana sekä muita oleellisia tietoja tietoturvallisuuden vuoksi.

Rivi ”<endpoint proto="javascript" address="./js/TADDM.js" payload="inbound-endpoint" />” kertoo, että MQL-kyselyyn TADDM-palvelimelta saatava vastaus ohjataan TADDM.js-nimiseen javascript-tiedostoon. Kyseisessä tiedostossa on käytettävissä payload-niminen muuttuja, joka sisältää TADDM-palvelimelta vastauksena saadun JSON-muotoisen datan. (Savolainen 2014.) TADDM.js-tiedosto kirjoitettiin seuraavaksi:

```
var obj = JSON.parse(payload);
if(obj.length > 0){
    var date = new Date();
    var datetime = date.getFullYear()
        + "-" + (date.getMonth() + 1)
        + "-" + date.getDate()
        + " " + date.getHours()
        + ":" + date.getMinutes()
        + ":" + date.getSeconds()
        + "." + date.getMilliseconds();
    var results = [];
    var myJSON = "";
    for(i = 0;i < obj.length;i++){
        var item = {
            "ModelObjectGuid":obj[i].guid,
            "FromGuid":obj[i].fromComputerSystem.guid,
            "FromLocationTag":obj[i].fromComputerSystem.locationTag,
            "FromDisplayName":obj[i].fromComputerSystem.displayName,
            "ToGuid":obj[i].toComputerSystem.guid,
            "ToLocationTag":obj[i].toComputerSystem.locationTag,
            "ToDisplayName":obj[i].toComputerSystem.displayName,
            "ForcedUpdate":datetime,
            "Active":"1"
        };
        results.push(item);
    }
    myJSON = JSON.stringify(results);
    payload = "EXECUTE [tietokannan_nimi].[dbo].[SetData]
@FullTableName='tietokannan_nimi.dbo.TADDMData',
@KeyFieldList='ModelObjectGuid',
@JSONData='" + myJSON + "';
EXECUTE tietokannan_nimi.dbo.DeactivateTADDMData;";
    payload;
}
```

TADDM.js-javascript-tiedoston koodi muuntaa TADDM-palvelimelta saatavan JSON-objektin ensin javascript-objekteiksi (var obj = JSON.parse(payload);). Sen jälkeen koodi käy objektit läpi for-silmukassa ja poimii item-nimiseen muuttujaan attribuutteja siten, että yksi item-muuttuja sisältää aina tiedon NetworkConnection-relaation molemmista palvelimista. Item-muuttujaan lisätään myös datetime-nimiseen muuttujaan koottu aikaleima, josta välitietokannassa tunnistetaan ne rivit jotka ovat saaneet päivityksen kyseisellä koodinsuorituskerralla. Tätä aikaleimaa hyödynnettiin kappaleessa 6.4 esitellyssä stored proceduressa. Item-objektit lisätään results-nimiseen taulukkoon, joka muutetaan lopuksi JSON-tekstiksi myJSON-nimiseen muuttujaan (myJSON = JSON.stringify(results);).

Enfo Oyj:n asiantuntijan mukaan (Kuutschin 2015) välitietokantaan voitiin syöttää JSON-muotoista dataa muotoilemalla payload-muuttuja seuraavaksi:

```
payload = "EXECUTE [tietokannan_nimi].[dbo].[SetData]
@FullTableName='tietokannan_taulun_nimi',
@KeyFieldList='ModelObjectGuid',
@JSONData='" + myJSON + "';
EXECUTE tietokannan_nimi.dbo.DeactivateTADDMData;";
```

Välitietokannassa oli olemassa valmis SetData-funktio, joka otti parametrikseen tietokannan kohdetaulun kokonimen (@FullTableName), avainkenttälistan (@KeyFieldList) ja JSON-muotoisen tekstin. Funktio osasi joko lisätä haluttuun tauluun uuden rivin tai päivittää olemassa olevaa riviä sen mukaan, löytyikö avainkentän (ModelObjectGuid) määrittämä rivi tietokannan taulusta vai ei. Samalla payload-muuttujaan määritettiin aiemmin luodun stored proceduren kutsu (EXECUTE tietokannan_nimi.dbo.DeactivateTADDMData;";) joka suoritetaan järjestyksessä ensimmäisen EXECUTE-lauseen jälkeen. (Kuutschin 2015.) Tämän koodin suorittamisen jälkeen jokainen yksittäinen välitietokannan taulun rivi sisälsi halutut tiedot NetworkConnection-relaation molemmista palvelimista. Lisäksi niiden rivien, joita ei enää löytynyt TADDM-palvelimen tietokannasta, Active-attribuutti oli merkitty nollaksi (0).

6.6 Tietojen hakeminen välitietokannasta ServiceNow-alustaan

Välitietokannassa olevat tiedot haettiin ServiceNow-alustaan ja tietojen perusteella tehtiin uusia relaatioita palvelimien välille. Relaatioita käsittelevä koodi kirjoitettiin siten, että seuraavilla suorituskerralla koodi poistaa jo olemassa olevat relaatiot, jos ne eivät enää ole voimassa (jos TADDM:n tietokanta ei enää sisällä kyseistä relaatiota).

Tiedot haettiin välitietokannasta käyttäen ServiceNow-alustan Scheduled Job -toimintoa, minkä avulla voidaan suorittaa mitä tahansa ServiceNow-alustan tukemaa koodia haluttuna ajankohtana (ServiceNow 2015a). Apuna käytettiin Enfo Oyj:n valmiita Script Include -koodikirjastoja, joiden avulla välitietokannan tiedot saatiin luettua. Script includeja käytetään yleisesti ServiceNow-alustassa varastoimaan palvelimella suoritettavaa Javascript-koodia (ServiceNow 2015c).

Tietojen hakeminen välitietokannasta ServiceNow-alustaan alkoi siten, että erääseen palvelimelle A asennetun Mule-ohjelman konfiguraatiotiedostoon lisättiin SQL-kyselylause. Tältä kyseiseltä palvelimelta A oli yhteys sekä välitietokannan sisältävälle palvelimelle että ServiceNow-alustaan. Palvelimelle A määritettävä kyselylause oli siis sellainen, minkä avulla haettiin halutut tiedot välitietokannasta. (Kuutschin 2015.) Konfiguraatiotiedostoon lisättiin seuraavat rivit:

```
<entry
    key="query:GetTADDMData"
    value="SELECT * FROM [tietokanta].[dbo].[taulu]
            WHERE [ModelObjectGuid]
                LIKE '{?ModelObjectGuid}'
                    AND Active = 1" />
```

Tämä mahdollisti sen, että ServiceNow-alustasta voitiin kutsua value-attribuutissa määritettyä SQL-kyselylausetta. Kun tätä kyselyä kutsuttaisiin, tekisi palvelin A kyselyn välitietokantaan ja palauttaisi tuloksen ServiceNow-alustaan javascript-objekteiksi (Kuutschin 2015).

SQL-kyselylauseen kutsu tehtiin ServiceNow-alustasta Enfo Oyj:n valmiita koodikirjastoja hyödyntäen. Kutsussa käytettiin metodia, joka otti parametrikseen query:n nimen - GetTADDMData – ja SQL:n SELECT-lauseeseen määritetyn muuttujan {?ModelObjectGuid}. Edellä mainittu SQL-kyselylauseen kutsu tehtiin ServiceNow-alustasta seuraavalla komennolla:

```
var ciList = WS.executeQuery('Database_location', 'GetTADDMData',
["ModelObjectGuid", "%"]);
```

Tässä komennossa muuttuja {?ModelObjectGuid} saa arvokseen prosenttimerkin (%), joten välitietokantaan tehtävä SQL-kysely muodostui seuraavaksi:

```
SELECT * FROM [tietokanta].[dbo].[taulu]
    WHERE [ModelObjectGuid] LIKE '%' AND Active = 1;
```

Tämä kysely hakee välitietokannasta kaikki sellaiset rivit, joiden Active-attribuutin arvo on 1. Jos jatkossa haluttaisiin hakea vain jokin tietty rivi välitietokannasta, voitaisiin executeQuery-metodiin määrittää prosenttimerkin paikalle jokin tietty ModelObjectGuid, jolloin komento palauttaisi välitietokannasta vain sen rivin, josta kyseinen ModelObjectGuid-attribuutti löytyisi (jos kyseisen rivin Active-attribuutti on 1).

6.7 Relaatioiden käsittely ServiceNow-alustassa

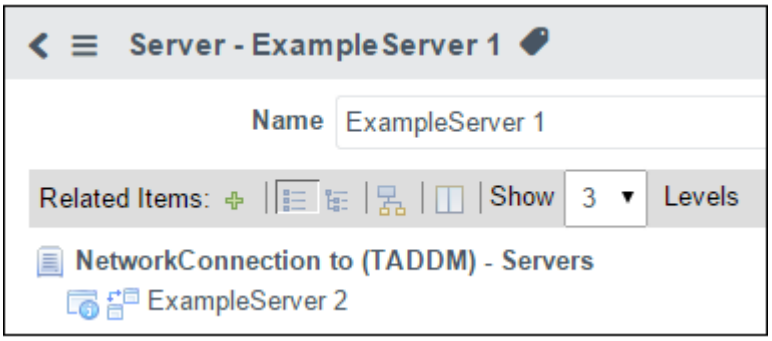
Välitietokannasta tiedon hakeva ja relaatioita käsittelevä koodi on nähtävissä kokonaisuudessaan erillisessä liitteessä (Liite 2). Koodia on paikoin muutettu salassapitovelvollisuuden vuoksi. Koodissa käytetään ServiceNow-alustan omaa GlideRecord-nimistä Java-luokkaa, jota voidaan käyttää javascript:n kanssa aivan kuin se olisi natiivi javascript-luokka. GlideRecord-luokkaa käytetään ServiceNow-alustassa tietokantaoperaatioihin SQL-lauseiden sijaan, joten luokan avulla voidaan:

- tehdä tietokantakysely
- hakea objekteja
- asettaa objekteille arvoja
- päivittää objekteja
- lisätä objekteja
- poistaa objekteja. (ServiceNow 2015b.)

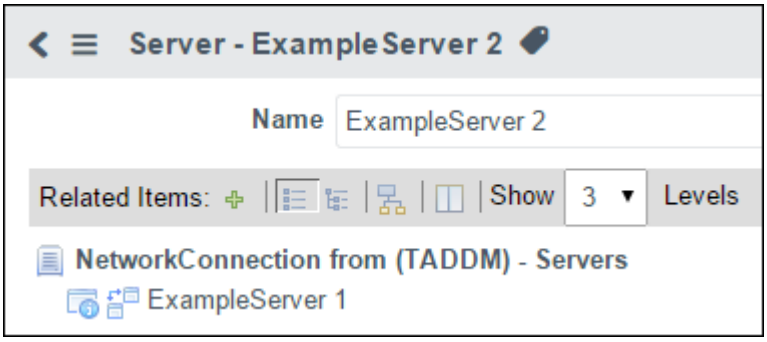
Vaikka ServiceNow-alusta tarjoaa oletuksena erityyppisiä relaatiovaihtoehtoja kahden konfiguraatio-objektin välille (ServiceNow 2015g), päätettiin TADDM-ohjelman tietojen perusteella luotaville relaatioille tehdä kokonaan uusi relaatiotyyppi. Relaatiotyyppi nimettiin seuraavasti: NetworkConnection from (TADDM)::NetworkConnection to (TADDM). Uusi relaatiotyyppi luotiin, koska relaatiotyypin alkuperä haluttiin tehdä selkeäksi. Jos Enfo Oyj:ssä työskentelevä asiantuntija esimerkiksi tutkii päivittäisessä työssään jonkin palvelimen välisiä relaatioita ServiceNow-alustan kautta, pystyisi hän näke-

mään suoraan relaatiotyypin nimestä mistä kyseinen relaatio on tullut ja millaista suhdetta relaatio esittää.

Seuraavissa kuvissa on esitetty ServiceNow-alustan käyttöliittymän näkymä kahden palvelimen välisestä NetworkConnection-relaatiosta. Ensimmäinen kuva (**Error! Reference source not found.**) näyttää, miltä relaatio näyttää sen palvelimen näkökulmasta josta relaatio saa alkunsa. Kuvasta nähdään, että ExampleServer 1:llä on NetworkConnection to (TADDM)-tyyppinen relaatio palvelimeen nimeltä ExampleServer 2. Jälkimmäinen kuva (**Error! Reference source not found.**) näyttää saman relaation toisesta suunnasta. Kuvasta nähdään, että ExampleServer 2:lla on NetworkConnection from (TADDM)-tyyppinen relaatio palvelimeen nimeltä ExampleServer 1.



KUVA 1. Palvelimien välinen relaatio ServiceNow-alustassa, esimerkki 1.



KUVA 2. Palvelimien välinen relaatio ServiceNow-alustassa, esimerkki 2.

Aluksi relaatioita käsittelevä koodi tutkii, löytyvätkö NetworkConnection-relaation molemmat osapuolet (palvelimet) ServiceNow-alustan tietokannasta. Jos relaation toisesta osapuolesta löytyy useampi kuin yksi objekti, ei oikeasta objektista voida olla varmoja ja niinpä tällöin ei tehdä mitään. Jos relaation molemmista osapuolista löytyy täsmälleen yksi objekti, tutkitaan seuraavaksi onko kyseisillä objekteilla jo olemassa NetworkConnection-relaatiota tai jotain muuta relaatiota, joka jo kertoo kyseisten objektien olevan yhteydessä toisiinsa. Jos haluttuja relaatioita ei objektien väliltä löydy, luodaan niiden välille uusi NetworkConnection-tyyppinen relaatio. Lopuksi poistetaan kaikki sellaiset NetworkConnection-relaatiot jotka eivät enää ole valideja. Kun relaatioita käsittelevä koodi on suoritettu, sisältää ServiceNow-alustan tietokanta viimeisimmät ajantasaiset relaatiotiedot TADDM-ohjelman löytämistä palvelimien välisistä NetworkConnection-relaatioista.

7 YHTEENVETO

Opinnäytetyön tavoitteena oli selvittää, saadaanko TADDM-konfiguraationhallintaohjelmasta siirrettyä konfiguraatiotietoa ServiceNow-palvelunhallinta-alustaan asiantuntijoiden käyttöön. Työn aikana TADDM-ohjelman ja ServiceNow-alustan välille luotiin onnistuneesti integraatiokanava, jota hyödyntäen TADDM-ohjelmasta saatiin siirrettyä palvelimien väliset relaatiotiedot ServiceNow-alustaan. Tietojen perusteella ServiceNow-alustassa luotiin joko uusia relaatioita tietokannassa olevien palvelinobjektien välille tai poistettiin sellaisia relaatioita, jotka eivät olleet enää päteviä. Opinnäytetyö tehtiin Enfo Oyj:lle osana yrityksessä käynnissä olevaa ”Konfiguraation- ja muutoksenhallinnan työkalu–POC (TADDM)” -nimistä projektia.

Integraatio toteutettiin Enfo Oyj:n käytäntöjen mukaisesti ja näin saatiin työelämän konkreettinen esimerkki integraation rakentamisesta. Vaikka opinnäytetyössä siirrettiin vain yhdenlaista informaatiota TADDM-ohjelmasta ServiceNow-alustaan, voi Enfo Oyj jatkossa käyttää samaa periaatetta ja integraatiokanavaa käytännössä minkä tahansa informaation siirtämiseen TADDM-ohjelmasta.

Opinnäytetyö oli opettavainen kokemus. Sen lisäksi että työn tekeminen vaati perehtymistä ITIL-käsitteeseen ja erityisesti palvelunhallintaan ja konfiguraationhallintaan, antoi se myös mahdollisuuden tutustua Enfo Oyj:n käytäntöihin integraatioiden toteutuksessa. Opinnäytetyön loppuraportin lisäksi Enfo Oyj:n käyttöön tuotettiin englanninkielinen dokumentaatio integraation toteutuksesta. Näitä dokumentteja ei kuitenkaan salassapitovelvollisuuden ja tietoturvallisuuden vuoksi julkaista opinnäytetyön yhteydessä.

LÄHTEET JA TUOTETUT AINEISTOT

ECMA INTERNATIONAL 2013. The JSON Data Interchange Format [verkkojulkaisu]. Ecma International. [Viitattu 2015-01-11.] Saatavissa: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

ENFO OYJ 2014a. Enfo lyhyesti [verkkojulkaisu]. Enfo Oyj. [Viitattu 2014-11-09.] Saatavissa: http://www.enfo.fi/enfo-group/enfo-lyhyesti/

IBM 2014a. Tivoli Application Dependency Discovery Manager. Class: net/NetworkConnection. IBM. [Viitattu 2015-01-03] Saatavissa: http://<TADDM_palvelimen_osoite>:9430/cdm/datadictionary/cdm/classes/net/NetworkConnection.htm (TADDM-ohjelman asennuspaketin mukana tuleva dokumentaatio, ei saatavissa ilmaiseksi)

IBM 2014b. Tivoli Application Dependency Discovery Manager 7.2.1. Application programming interface overview [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2014-12-13.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_api_overview.html

IBM 2014c. Tivoli Application Dependency Discovery Manager 7.2.1. Introducing the Common Data Model [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2014-12-13.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_datamodel_introducing.html

IBM 2014d. Tivoli Application Dependency Discovery Manager 7.2.1. Making REST calls with a Web browser [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2014-12-13.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/t_cmdbsdk_restapi_browser.html

IBM 2014e. Tivoli Application Dependency Discovery Manager 7.2.1. Overview of the Software Developer Kit (SDK) [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2014-13-12.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_sdkoverview.html

IBM 2014f. Tivoli Application Dependency Discovery Manager 7.2.1. REST API overview [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-02.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_restapi_overview.html

IBM 2014g. Tivoli Application Dependency Discovery Manager 7.2.1. Undestanding the Common Data Model [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2014-13-12.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.2.1/com.ibm.taddm.doc_721/SDKDevGuide/c_cmdbsdk_understandingdatamodel.html

IBM 2014h. Tivoli Application Dependency Discovery Manager 7.3.0. JSON format overview [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-03.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/c_cmdbsdk_json_overview.html

IBM 2014i. Tivoli Application Dependency Discovery Manager 7.3.0. Model Query Language overview [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-03.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/c_cmdbsdk_mql_introducing.html

IBM 2014j. Tivoli Application Dependency Discovery Manager 7.3.0. MQL query service [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-03.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/r_cmdbsdk_restapi_mqlquery.html

IBM 2014k. Tivoli Application Dependency Discovery Manager 7.3.0. Querying model objects using the REST API [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-03.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/t_cmdbsdk_restapi_query.html

IBM 2014l. Tivoli Application Dependency Discovery Manager 7.3.0. TADDM Data Dictionary [verkkojulkaisu]. IBM Knowledge Center. [Viitattu 2015-01-03.] Saatavissa: http://www-01.ibm.com/support/knowledgecenter/SSPLFC_7.3.0/com.ibm.taddm.doc_7.3/SDKDevGuide/c_cmdbsdk_datadict.html

IBM 2014m. IBM Tivoli Application Dependency Discovery Manager Capabilities and Best Practices [verkkojulkaisu]. IBM. [Viitattu 2014-11-01.] Saatavissa: http://www.redbooks.ibm.com/redbooks/pdfs/sg247519.pdf

KUUTSCHIN, Eero 2015-05-01. Specialist. [Haastattelu.] Kuopio: Enfo Oyj.

OFFICE OF GOVERNMENT COMMERCE 2010a. Continual Service Improvement. Third Impression. United Kingdom: TSO (The Stationery Office).

OFFICE OF GOVERNMENT COMMERCE 2010b. Service Design. Third Impression. United Kingdom: TSO (The Stationery Office).

OFFICE OF GOVERNMENT COMMERCE 2010c. Service Operation. Third Impression. United Kingdom: TSO (The Stationery Office).

OFFICE OF GOVERNMENT COMMERCE 2010d. Service Strategy. Third Impression. United Kingdom: TSO (The Stationery Office).
OFFICE OF GOVERNMENT COMMERCE 2010e. Service Transition. Third Impression. United Kingdom: TSO (The Stationery Office).

OFFICE OF GOVERNMENT COMMERCE 2007. The Official Introduction to the Itil Service Lifecycle. United Kingdom: TSO (The Stationery Office).

SAVOLAINEN, Tero 2014-12-19. Senior Specialist. [Haastattelu.] Kuopio: Enfo Oyj.

ServiceNow 2015a. Creating a Scheduled Job [verkkojulkaisu]. ServiceNow. [Viitattu 2015-01-10]. Saatavissa: http://wiki.servicenow.com/index.php?title=Creating_a_Scheduled_Job

ServiceNow 2015b. GlideRecord [verkkojulkaisu]. ServiceNow. [Viitattu 2015-02-08]. Saatavissa: http://wiki.servicenow.com/?title=GlideRecord

ServiceNow 2015c. Script Includes [verkkojulkaisu]. ServiceNow. [Viitattu 2015-01-10]. Saatavissa: http://wiki.servicenow.com/index.php?title=Script_Includes

ServiceNow 2015d. ServiceNow Company Overview [verkkojulkaisu]. ServiceNow. [Viitattu 2015-02-07]. Saatavissa: http://www.servicenow.com/content/dam/servicenow/documents/corporate/ServiceNow_Company_Overview.pdf

ServiceNow 2015e. ServiceNow Configuration Management [verkkojulkaisu]. ServiceNow. [Viitattu 2015-02-08]. Saatavissa: http://www.servicenow.com/products/it-service-automation-applications/configuration-management.html

ServiceNow 2015f. ServiceNow Products [verkkojulkaisu]. ServiceNow. [Viitattu 2015-02-08]. Saatavissa: http://www.servicenow.com/products.html

ServiceNow 2015g. Suggested Relationships [verkkojulkaisu]. ServiceNow. [Viitattu 2015-02-08]. Saatavissa: http://wiki.servicenow.com/index.php?title=Suggested_Relationships

LIITE 1. MQL QUERY DISMANTLED

URL - scheme//hostname:port/rest/model/MQLQuery

where:

- scheme – Can be HTTP or HTTPS
- hostname – Hostname or IP address of the TADDM server
- port – Specify 9430 for HTTP or 9431 for HTTPS

HTTP methods

- GET

Parameters

- depth=value – Optional, default value = 1. Defines the depth of the query.
  - Note: If the depth is greater than 1, the query can return a large result set which can cause low-memory conditions on the TADDM server. It is recommended that you specify fetchSize=1 and use multiple queries to scroll through the data one position at a time. You can find sample programs in the $COLLATION_HOME/sdk/examples/rest directory on the TADDM server where this technique is used.
- feed={json|xml} – Optional. The data is returned in the format specified by this parameter. Optional parameter. If not specified, the server uses the format specified by the HTTP Accept header. If that is not specified, data is returned in the JSON format.
- fetchSize=value – Optional, default value = 1. Defines the maximum number of objects to be returned.
- longClassName={true|false} – Optional, default value = false. Valid only for the JSON format. If true, all model object class names in the output must be specified using the fully qualified form (for example, com.collation.platform.model.topology.sys.ComputerSystem).
- mssGuid=value – Optional. Defines the GUID value of the management software system (MSS) associated with the object.
- position=value – Optional, default value = 1. Defines the starting position in the result set for the objects you want returned from the query. If the position specified is greater than the total number of objects in the result set, no objects are returned.
- query=value – Mandatory. The MQL query string.
  - Note: To avoid memory and performance problems, write a query that selects only the columns you need.

Returns

- The server returns HTTP return code 200 if the query is successful. The server also returns the query result data either in JSON or XML format as specified in parameters section. If no data is returned, the result data will be either an empty JSON array or an empty XML document.

- If the TADDMQueryComplete pragma header of the returned data is false, it means that more query results are available and if it is true, it means that all results have been returned.

Example

- This is an example that queries model object data using the MQL query "select displayName,OSRunning.OSName from ComputerSystem"
- http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,OSRunning.OSName%20from%20ComputerSystem&position=2&fetchSize=2&feed=xml&depth=2&position=4

(IBM 2014j.)

LIITE 2. VÄLITIETOKANNASTA TIEDOT HAKEVA JA RELAATIOT KÄSITTELEVÄ KOODI SERVICENOW-ALUSTASSA

```
//include webservices library
gs.include('Webservices');

//include execution library
gs.include('Exec');

//create a new instance of webservices
var WS = new Webservices();

//create a new execution instance
var exec = new Exec();

//holds abbreviations that we need to check
var abbreviationMap = {};

//sys_id of the new relationship to be created between servers
//NetworkConnection from (TADDM)::NetworkConnection to (TADDM)
var newRelationType = '';

//holds existing relations that we need to check between servers
var existingRelationsToCheck = [];

//collects and holds relations that are valid.
//At the end of the script, relations that are NOT in this array
//(old relations) will be deleted
var validRelations = [];

//enable/disable logging
var debugLogging = true;

//if false, test-data from the end of the script is used.
var useRealData = false;

//run main()
exec.run("TADDM_script", "Scheduled Script", this.main);

// ====================================================
function main() {

    // populates abbreviationMap and existingRelationsToCheck
    initialize();

    if(useRealData){
        // ============load data from Database=========
        var ciList = WS.executeQuery('map key',
                                     'GetTADDMData',["ModelObjectGuid", "%"]);
        if(useRealData)
            exec.log("CI count:" + ciList.length);
    }else{
        var ciList = getTestData();
    }

    if(ciList.length > 0){
        for (var i = 0; i < ciList.length; i++) {

            // exec.progress() returns false if this script has forbid flag
set.
            if(!exec.progress(i+1, ciList.length, 400)) {
                exec.setMessage("Interrupted (forbid). " + i
+ " CI's processed.");
            return "forbid";
        }
```

```
//get Networkconnection FROM-server record
        var fromServer = GetServerParseAssetTag(ciList[i].FromLocationTag,
ciList[i].FromDisplayName);

        if(fromServer){
            //get Networkconnection TO-server record
            var toServer = GetServerParseAssetTag(ciList[i].ToLocationTag,
ciList[i].ToDisplayName);

            if(toServer){
                //check if servers already have TADDM-relation
                var relationExists = checkExistingNewRelation(fromServer,
toServer);

                if(!relationExists){
                    //check if servers have one of the relations defined in
                    //"existingRelationsToCheck"-array (from-to)
                    relationExists = checkExistingRelations(fromServer,
toServer);

                    if(!relationExists){
                        //check if servers have one of the relations defined
in
                        //"existingRelationsToCheck"-array (to-from)
                        relationExists = checkExistingRelations(toServer,
fromServer);

                        if(!relationExists){
                            //create new TADDM-relation if no existing rela-
tions were found
                            createNewRelationship(fromServer, toServer);
                        }
                    }
                }
            }else{
                if(debugLogging)
                    exec.log("Could not find exactly one instance of
TO-server with locationTag: "
                        + ciList[i].ToLocationTag + "and displayName: " +
ciList[i].ToDisplayName)
                }
            }else{
                if(debugLogging)
                    exec.log("Could not find exactly one instance of FROM-server
with locationTag: "
                        + ciList[i].FromLocationTag + "and displayName: " +
ciList[i].FromDisplayName)
            }
        }
        //delete old TADDM-relations
        deleteOldRelations();
}else{
        exec.log("========Zero rows of data to process!=========");
}
}
```

```
// ====================================================
// GetServerParseAssetTag()
//
function GetServerParseAssetTag(LocationTag, DisplayName) {
    var howManyFound = 0;
    var debugString = "";
    var assetTag = "";
    var serverFound;
    var server = new GlideRecord('cmdb_ci_server');

    //look for every abbreviation defined in abbreviationMap
    for(var i = 0;i<abbreviationMap[LocationTag].length;i++){
        server.initialize();
        assetTag = abbreviationMap[LocationTag][i] + "|" +
                    DisplayName.toString().split('.')[0] + "|";
        server.addQuery('asset_tag', 'STARTSWITH', assetTag);
        server.addQuery('u_active', 1);
        server.query();

        howManyFound += server.getRowCount();
        debugString += "\nFound " + server.getRowCount().toString()
                        + " server(s) with asset tag: " + assetTag;

        //how many have we found so far?
        if(howManyFound <= 1){

            //have we found a server in this loop?
            if(server.getRowCount() > 0){

                //ONLY one server found, save it for later
                server.next();
                serverFound = server;
            }
        }else{

            //multiple servers found, break out of loop
            i = abbreviationMap[LocationTag].length;
        }
    }

    //if only one instance of server found
    if(howManyFound == 1){
        if(debugLogging)
            exec.log("GetServerParseAssetTag() : SUCCESS :"
+ debugString);
        return serverFound;
    }else{
        if(debugLogging)
            exec.log("GetServerParseAssetTag() : FAIL :" + debugString);
        return null;
    }
}
```

```
// ====================================================
// checkExistingRelations()
//
function checkExistingRelations(parentServer, childServer){
    var exists = false;
    var relation = new GlideRecord('cmdb_rel_ci');
    for(var i = 0;i < existingRelationsToCheck.length && !exists;i++){
        relation.initialize();
        relation.addQuery('type', existingRelationsToCheck[i]);
        relation.addQuery('parent', parentServer.sys_id);
        relation.addQuery('child', childServer.sys_id);
        relation.query();
        if(relation.next()){
            if(debugLogging){
                exec.log("Relation exists between "
                    + parentServer.asset_tag + " and " +
childServer.asset_tag);
            }
            exists = true;
        }
    }
    return exists;
}

// ====================================================
// checkExistingNewRelation()
//
function checkExistingNewRelation(parentServer, childServer){
    var exists = false;
    var relation = new GlideRecord('cmdb_rel_ci');
    relation.addQuery('type', newRelationType);
    relation.addQuery('parent', parentServer.sys_id);
    relation.addQuery('child', childServer.sys_id);
    relation.query();
    if(relation.next()){
        validRelations.push(relation.sys_id);
        if(debugLogging){
            exec.log("TADDM Relation exists between "
                + parentServer.asset_tag + " and " + childServer.asset_tag);
        }
        exists = true;
    }
    return exists;
}


// ====================================================
// createNewRelationship()
//
function createNewRelationship(fromServer, toServer){
    var newRelation = new GlideRecord('cmdb_rel_ci');
    newRelation.initialize();
    newRelation.parent = fromServer.sys_id;
    newRelation.child = toServer.sys id;
    newRelation.type = newRelationType;
    newRelation.insert();
    validRelations.push(newRelation.sys_id);
    if(debugLogging){
        exec.log("New relation created between "
            + fromServer.asset_tag +" and " + toServer.asset_tag);
    }
}
```

```
// ====================================================
// deleteOldRelations()
//
function deleteOldRelations(){
    var relations = new GlideRecord('cmdb_rel_ci');
    relations.addQuery('type', newRelationType);
    relations.addQuery('sys_id', 'NOT IN', validRelations);
    relations.query();

    if(debugLogging){
        exec.log("Found " + relations.getRowCount()
            + " existing TADDM relations that are not valid anymore.");
    }

    while(relations.next()){
        if(debugLogging){
            exec.log("Deleting relationship between :  "
                + relations.parent.asset_tag + " AND : "
                    + relations.child.asset_tag);
        }
        relations.deleteRecord();
    }
}

// ====================================================
// initialize()
//
function initialize()
{
    //for example, 'company1' is the locationTag in TADDM,
    //but it can be "COM1" or "COMP1" in ServiceNow
    abbreviationMap = {
        'company1': [ "COM1", "COMP1" ],
        'company2': [ "COM2", "COMP2" ],
        'company3': [ "COM3" ]
    };

    //existence of these relations are checked
    //between servers before TADDM-relation is created
    existingRelationsToCheck =
    [
    '', //Connected by :: Connects
    '' //Connects to :: Connected by
    ];
}

function getTestData(){
    //=========actual data : can be used for testing
    var testObject =
    [
    {
        "FromLocationTag" : "XX",
        "FromDisplayName" : "test1.100.local",
        "ToLocationTag" : "YY",
        "ToDisplayName" : "test2.100.local",
    },
    {
        "FromLocationTag" : "XXX",
        "FromDisplayName" : "test3.100.local",
        "ToLocationTag" : "YYY",
        "ToDisplayName" : "test4.100.local",
    }
    ];
    return testObject;
}
```